

# Hypergraph Partitioning for Computing Matrix Powers

Erin Carson, James Demmel, and Nicholas Knight

29 October 2010

**Motivation** Krylov Subspace Methods (KSMs) are a class of iterative algorithms commonly used in scientific applications for solving linear systems, eigenvalue problems, singular value problems, and least squares. Standard KSMs are communication-bound, due to a sparse matrix vector multiplication (SpMV) in each iteration. This motivated the formulation of *Communication-Avoiding* KSMs, which remove the communication bottleneck to increase performance. A successful strategy for avoiding communication in KSMs uses a matrix powers kernel that exploits locality in the graph of the system matrix  $A$ . The matrix powers kernel computes  $k$  basis vectors for a Krylov subspace (i.e.,  $\mathcal{K}_k(A, v) = \text{span}\{v, Av, \dots, A^{k-1}v\}$ ) reading  $A$  only once. Since a standard KSM reads  $A$  once per iteration, this approach effectively reduces the communication cost by a factor of  $k$  [7, 8].

The current implementation of the matrix powers kernel [8] partitions the matrix  $A$  given the computed dependencies using graph partitioning of  $A + A^T$ . However, the graph model inaccurately represents the communication volume in SpMV and is difficult to extend to the case of nonsymmetric matrices. A hypergraph model remedies these two problems for SpMV [2, 5, 3]. The fundamental similarity between SpMV and the matrix powers kernel motivates our decision to pursue a hypergraph communication model.

**Contribution** We construct a hypergraph that encodes the matrix powers communication, and prove that a partition of this hypergraph corresponds exactly to the communication required when using the given partition on the rows of  $A$ . Although the hypergraph construction represents an additional preprocessing cost, such a cost is justified when the resulting partition requires significantly less communication than a partition obtained using a naive or graph partitioning approach. We evaluate this tradeoff for various classes of matrices, and provide guidance in selecting the optimal partitioning method in terms of preprocessing cost and communication required in the resulting partition. Additionally, we evaluate the effectiveness of various heuristics and sparsification strategies for reducing the cost of constructing and partitioning the hypergraph for matrix powers.

Using the PaToH hypergraph partitioning software [4] and matrices from the U.F. Collection [6], we achieve up to an 80% reduction in total (expand) communication volume over the graph partitioning approach, the largest improvements for structurally unsymmetric matrices (e.g., west1505). In general, the matrix powers kernel shows the most benefit on well-structured (bounded aspect ratio) matrices. We demonstrate that our techniques will improve matrix powers performance for classes of graphs such as physical meshes that are structurally unsymmetric.

**A Hypergraph Model for Matrix Powers** Our formulation is based on the column-net model, previously described in literature for SpMV [2, 5, 3]. The column-net model was chosen because the matrix powers kernel requires a rowwise partition, ensuring no communication occurs during the computation. We first demonstrate correctness of this formulation for  $y = A^k \cdot x$  where  $k = 1$ , and then consider the case where  $k > 1$ . The column-net model still holds for the computation of  $y = A^k \cdot x$ , provided that we find the column-nets for matrix  $A^k$ .

However, matrix powers does not simply calculate an SpMV for some power of  $A$ . In fact, matrix powers performs an SpMV for all powers of  $A$  up to  $k$ , and then returns the vectors  $[x, Ax, \dots, A^k x]$ . In general, each power of  $A$  expresses a different set of dependencies so it is insufficient to evaluate the dependencies for  $A^k$  and claim that those encompass all communication in matrix powers. We show how to combine the dependencies for matrix powers  $A^1$  through  $A^k$  in a way to account for all dependencies. We perform this combination using a union operation, since once a dependency has been accounted for, the communicated value is available locally (without subsequent communication) for all iterations. These new nets, which we call *k-level column-nets*, represent the cumulative dependencies.

We form a hypergraph  $H_k$  with the same vertices as  $H$ , above, but now use the  $k$ -level column-nets. We have encapsulated dependencies by nets in  $H_k$  so that the cost of a  $P$ -way partition in  $H_k$  tells us the communication that will occur in the execution of the matrix powers kernel. It is worth noting that minimizing communication in the matrix powers kernel is NP-Hard since the transformation  $[A, k, x] \rightarrow H_k$  can be performed in polynomial

number of steps, giving a polynomial-time reduction from minimizing communication in the matrix powers kernel to finding the minimum cost cut of  $H_k$ .

**Using Heuristics to Reduce Preprocessing Cost** Constructing the full  $k$ -level column nets adds a significant cost to the preprocessing algorithm, especially for large values of  $k$ . We evaluate the effectiveness of the following strategies (and combinations of these strategies) for reducing the algorithmic cost of both constructing and partitioning our hypergraph for matrix powers:

- Sparsification of the input matrix
- Dropping large nets from consideration during partitioning
- Approximating  $k$ -level column-nets with  $k = 1 +$  iterative swapping [9]

**Current Work** Current work involves considering the costs of moving the redundant entries of the input vector  $x$  as well as the rows of matrix  $A$  independently. That is, our previous formulation assumes a unit cost for each data dependency, when it would be more accurate to weight the  $k$ -level dependencies with unit cost  $w_i = 1$ , and the 1- through  $(k - 1)$ -level dependencies by a weight  $w_i = 1 + nnz_i$ , where  $nnz_i$  is the number of nonzeros in matrix row  $A_i$ . This would be formulated as a multi-constraint hypergraph partitioning problem, discussed in [1] in the context of SpMV.

Additionally, ongoing work involves exploring the effectiveness of such a strict communication-avoiding approach in practice. The communication costs between cores are far less expensive than, for example, between nodes in a network. Other partitioning schemes, like the 2D decompositions evaluated in [5], would require communication at each step of the matrix powers kernel (since columns are partitioned as well), but have greater flexibility in reducing the overall volume of communication. We believe this flexibility may play a role in an optimal organization of the algorithm in a shared-memory environment, and would also provide robustness in pathological cases like a dense row in  $A$ .

## References

- [1] C. Aykanat, B.B. Cambazoglu, and B. Ucar. Multi-level direct  $k$ -way hypergraph partitioning with multiple constraints and fixed vertices. *Journal of Parallel and Distributed Computing*, 68(5):609–625, 2008.
- [2] U.V. Catalyurek and C. Aykanat. *Decomposing irregularly sparse matrices for parallel matrix-vector multiplication*, In *Parallel Algorithms for Irregularly Structured Problems. Lecture Notes in Computer Science, Vol. 1117/1996*, pages 75–86. Springer Berlin / Heidelberg, 1996.
- [3] U.V. Catalyurek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse- matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.
- [4] U.V. Catalyurek and C. Aykanat. Patoh: A multilevel hypergraph partitioning tool, version 3.0. Bilkent University, Department of Computer Engineering, Ankara, 06533 Turkey., 1999.
- [5] U.V. Catalyurek, C. Aykanat, and B. Ucar. On two-dimensional sparse matrix partitioning: Models, methods, and a recipe. *SIAM J. Sci. Comput.*, 32(2):656–683, 2010.
- [6] T. Davis. University of florida sparse matrix collection. NA Digest, 92, 1994.
- [7] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick. Avoiding communication in computing krylov subspaces. Technical Report No. UCB/EECS-2007-123, 2007.
- [8] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick. Minimizing communication in sparse matrix solvers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, 2009.
- [9] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partition. *Proc. 19th IEEE Design Automation Conference*, pages 175–181, 1982.